

A workflow for building surface-based reservoir models using NURBS curves, coons patches, unstructured tetrahedral meshes and open-source libraries

Citation for published version:

Zhang, Z, Yin, Z & Yan, X 2018, 'A workflow for building surface-based reservoir models using NURBS curves, coons patches, unstructured tetrahedral meshes and open-source libraries', *Computers and Geosciences*, vol. 121, pp. 12-22. <https://doi.org/10.1016/j.cageo.2018.09.001>

Digital Object Identifier (DOI):

[10.1016/j.cageo.2018.09.001](https://doi.org/10.1016/j.cageo.2018.09.001)

Link:

[Link to publication record in Heriot-Watt Research Portal](#)

Document Version:

Peer reviewed version

Published In:

Computers and Geosciences

Publisher Rights Statement:

© 2018 Elsevier B.V.

General rights

Copyright for the publications made accessible via Heriot-Watt Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy

Heriot-Watt University has made every reasonable effort to ensure that the content in Heriot-Watt Research Portal complies with UK legislation. If you believe that the public display of this file breaches copyright please contact open.access@hw.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

A Workflow for Building Surface-Based Reservoir Models Using NURBS Curves, Coons Patches, Unstructured Tetrahedral Meshes and Open-Source Libraries

Zhao Zhang*

Institute of Petroleum Engineering, Heriot-Watt University, Edinburgh, EH14 4AS, UK

Zhen Yin

Department of Geological Sciences, Stanford University, CA, 94305

Xia Yan

*School of Petroleum Engineering, China University of Petroleum (East China),
Shandong, China*

Abstract

Surface-based models have been built to represent complex reservoir geometries. This paper presents a workflow for building surface-based reservoir models using NURBS curves, Coons patches and unstructured tetrahedral volume meshes. Surfaces are created as Coons patches based on NURBS curves. The surface mesh of the entire model is hybrid consisting of quadrilaterals and triangles. Geological regions are represented as volumes bounded by surfaces. Unstructured tetrahedral meshes are built to adapt to the bounding surfaces. Well configurations of location and geometry are particularly flexible, facilitated by mesh adaptation. All libraries for curve, surface and mesh generation are open-source. They are free-of-charge for non-commercial uses. The workflow provides a flexible alternative to commercial software packages for building surface-based models and unstructured meshes. The workflow is validated by simulating two-phase immiscible displacement and comparing

*Corresponding author

Email address: `zhao.zhang@hw.ac.uk` (Zhao Zhang)

to the analytical solution.¹

Keywords: NURBS, Coons patch, unstructured tetrahedral mesh, surface-based reservoir modelling, open-source

1. Introduction

The conventional practice in hydrocarbon reservoir modelling is based on corner-point grids (CPG). As uncertainty is reduced in an oil or gas field with more exploration and production data becoming available, usually only a single detailed, history-matched, full-field, base-case model is maintained (Bentley, 2016). The model is then upscaled for simulation. The main disadvantages of this practice include that it is difficult and very time-consuming to change the underlying geological concepts of a detailed model; hard to explore a range of reservoir prototypes; the geometry of the geological architectures is grid dependent and CPG limit the geometric complexities that can be captured.

Another approach is to build surface-based reservoir models (SBRM) such that the model can be grid-independent (Bentley and Ringrose, 2017). Surfaces that separate the model into different volumes or regions are built in the reservoir prototyping phase (Cavero et al., 2016) and are enriched as new data become available. The idea of SBRM has been applied in software packages such as GSI3D, Move and Gocad.

Volumes bounded by the surfaces need to be meshed for simulation. In general, there are two types of meshes which are structured (e.g. Cartesian, curvilinear or corner-point grids) and unstructured meshes (e.g. unstructured tetrahedral, polyhedral or hybrid meshes). Compared to structured grids, unstructured meshes offer a more flexible way to adapt to complex geometries automatically such that meshes conform to model architectures rather than model architectures conforming to meshes (Milliotte and Matthäi, 2014; Kumar et al., 2016). This explains the popularity of unstructured meshes in computer aided engineering (CAE) or design (CAD). There are a number

¹The contribution of Zhao Zhang is the development and implementation of the workflow and the writing of the paper. The contribution of Zhen Yin is help with the introduction of the paper and analysis of how to build NURBS curves from seismic, outcrop and sparse data. The contribution of Xia Yan is help with the design of test cases. All authors have approved the paper and agree its submission.

of commercial or free 3D unstructured mesh generators for CAE or CAD, including ANSYS Meshing, STAR-CCM+, GMSH, Gocad FEMC, Abaqus and Pointwise. A comparative study of workflows using some of these software packages for unstructured mesh generation with SBRM is reported in Zehner et al. (2015).

Question is: can we build SBRM and unstructured meshes using freely available and open-source codes? The answer is yes. In this paper, we will present a workflow of constructing SBRM and unstructured meshes based on open-source codes and libraries. A model builder written in Matlab and a mesh generator written in C++ are developed to implement the workflow. All libraries are incorporated internally for efficiency and we only need files for exporting data from the model builder to the mesh generator.

Non-uniform Rational Basis Spline (NURBS) is commonly used in the computer aided design community for describing curves and surfaces (Piegl and Tiller, 2012). It has also been applied to geological modelling (Zhong et al., 2006). Here we use NURBS for generating curves and bilinearly blended Coons patches (BBCP) for surfaces. The approach is termed NURBS-Curves-BBCP. A Coons patch is a type of manifold parametrization used in computer graphics (Farin and Hansford, 1999). We only need four curves to generate a BBCP. The functions for generating NURBS and BBCP are available in the open-source Matlab codes NURBS Toolbox.

To make reservoir models reliable, reservoir geological structures interpreted from 2D/3D seismic and outcrop observations can be used to guide the generation of curves and surfaces (Novakovic et al., 2002; Ruiiu et al., 2016; Colombera et al., 2018). Besides, sparse data observations such as well data (e.g. horizon picks) can also be assimilated to the NURBS curves to further correct the reservoir models. Compared to NURBS surfaces Ruiiu et al. (2016), the NURBS-Curves-BBCP approach requires less inputs (only curves) to generate 3D models. Another important advantage of the NURBS-Curves-BBCP approach is that it enables more flexible update of reservoir models when new structure patterns are discovered during the reservoir development. For example, it is found that 4D seismic monitoring acquired after the reservoir production can detect new reservoir structures which are initially not interpreted from the 3D seismic (Yin et al., 2015). It therefore requires the reservoir model structures to be updated. But the update of reservoir structural models can be complex and very time consuming if using the conventional CPG modelling. This problem is avoided in NURBS by simply updating the curves.

65 The connection between Coons patches and unstructured volume mesh
66 generation is a surface mesh of the entire model. The surface mesh needs to
67 adapt to all intersection curves between surfaces to be water-tight, which is
68 a requirement for generating a volume mesh for simulation. However, there
69 is no open-source codes for generating a 3D surface mesh on a given set of
70 Coons patches.

71 In the current study, a hybrid surface mesh is built for the entire model.
72 Structured logically Cartesian grids are constructed on Coons patches. Inter-
73 section curves on each vertical bounding surface are discretised into a Planar
74 Straight Line Graph (PSLG) which is read into the open-source C library Tri-
75 angle (Shewchuk, 1996, 2002) to generate adaptive triangular meshes . Then
76 the meshes on Coons patches and vertical bounding surfaces are connected
77 to form a surface mesh of the entire model.

78 The open-source C++ library TetGen (Si, 2015) is employed for un-
79 structured volume mesh generation. The hybrid surface mesh is stored as
80 a Piecewise Linear Complex (PLC) which is the input format for TetGen.
81 Tetrahedral meshes adapt to bounding surfaces in the sense that the surfaces
82 can be represented as the connected facets of tetrahedral elements. Quality
83 mesh generation according to constraints and local refinement are available
84 in TetGen. An alternative for TetGen could be CGALmesh which is part
85 of the CGAL library. However as reported in Si (2015), TetGen is more
86 computationally efficient than CGALmesh. Both TetGen and CGALmesh
87 allows compiling them as internal libraries. As CGALmesh is dependent on
88 other libraries in CGAL, using CGALmesh requires incorporating all these
89 libraries which is less convenient than using TetGen.

90 Some other studies for unstructured mesh generation on 3D geological
91 models can be found in literature. Wang et al. (2017) presented an ap-
92 proach for generating Delaunay discretisation for 3D discrete fracture net-
93 works where they generate Delaunay triangular meshes on surfaces and then
94 use TetGen to create adaptive tetrahedral meshes. However, all surfaces in
95 their study are flat while curved surfaces are considered in our study. Pellerin
96 et al. (2017) developed a library for reading and writing surface and volume
97 meshes in various formats to help researchers interact with different software
98 packages of mesh generation, simulation and visualisation. The library can
99 be used with our workflow to convert volume meshes generated by TetGen
100 into formats compatible with different software packages.

101 This paper is organised as follows. First, open-source codes NURBS
102 Toolbox, Triangle and TetGen are reviewed. Second, the workflow of build-

ing SBRM, hybrid surface meshes and adaptive volume meshes is discussed. Third, mesh post-processing techniques are reviewed. Finally, test cases for our workflow are presented.

2. Review of Open-Source Libraries

2.1. NURBS Toolbox

NURBS stands for Non-uniform Rational Basis Spline. It is the standard for describing and modelling curves and surfaces in CAD and computer graphics. For an introduction to the mathematical theories of NURBS, please see Rogers (2000) and Piegl and Tiller (2012). A NURBS curve is described by a list of control points and a knot vector. A curve is represented as a series of polynomials defined by the control points. The knots determine the start and end locations of the polynomials. The number of control points should be at least equal to the order of the curve, while the length of the knot vector is the sum of the number of control points and the order of the curve. Knots are defined in the parametric space. In the current study, the open-source Matlab library NURBS Toolbox is employed (Spink, 2010) for both NURBS curves and Coons patches. For a curve, the range of the values of knots is $[0, 1]$ in NURBS Toolbox. Here, we use quadratic curves.

A Coons patch is a type of manifold parametrization (Farin and Hansford, 1999). Given four boundary curves, a parametric surface can be generated as a bilinearly blended Coons patch that interpolates to the curves. The parametric definition of a surface is

$$\mathbf{x} = (x, y, z) = f(u, v) , \quad (1)$$

where (x, y, z) is the location in 3D, u and v are the parametric coordinates and f is a mapping function. In NURBS Toolbox, the range of u and v is $[0, 1]$ which does not contain information of the aspect ratio (length of longest side over that of shortest side) of a 3D surface. Four boundary curves are

$$f(0, v), \quad f(1, v), \quad f(u, 0), \quad f(u, 1), \quad (0 \leq u \leq 1, 0 \leq v \leq 1). \quad (2)$$

The definition of a bilinearly blended Coons patch is (Farin and Hansford, 1999)

$$\begin{aligned}
f(u, v) = & (1 - u)f(0, v) + uf(1, v) \\
& + (1 - v)f(u, 0) + vf(u, 1) \\
& - [1 - u, v] \begin{bmatrix} f(0, 0) & f(0, 1) \\ f(1, 0) & f(1, 1) \end{bmatrix} \begin{bmatrix} 1 - v \\ v \end{bmatrix}, \tag{3}
\end{aligned}$$

131 which is the sum of two linear interpolations minus a bilinear interpolation.

132 2.2. *Triangle*

133 The open-source C library Triangle (Shewchuk, 1996, 2002) is adopted for
134 Delaunay triangulation in 2D. Triangle has been applied in various areas and
135 has been cited several thousand times (De Berg et al., 2000; Tu et al., 2018;
136 Welsh and Mainland, 2004). It can generate exact Delaunay triangulations
137 (DT), constrained DT, conforming DT, Voronoi diagrams, high-quality tri-
138 angular meshes and refine existing triangulations (Chew, 1993). Constraints
139 on angles and areas of triangles can be implemented for high-quality mesh
140 generation. Both incremental and divide-and-conquer algorithms for DT are
141 available (Guibas and Stolfi, 1985). Besides, the maximum allowed number
142 of Steiner points can be precisely controlled during triangulation.

143 The input for Triangle is a Planar Straight Line Graph (PSLG) that
144 is a collection of edges and associated vertices. The constraints for mesh
145 generation on each surface are four boundary curves and internal intersection
146 curves. The edges of a PSLG cannot be subdivided in a constrained DT, but
147 may be subdivided in a conforming DT. In other words, Steiner points are
148 allowed in a conforming DT but not a constrained DT. Yet, some triangles
149 in a constrained DT might not be Delaunay.

150 2.3. *TetGen*

151 The open-source TetGen library in C++ (Si, 2015) is adopted for un-
152 structured tetrahedral mesh generation. TetGen can be used as either a
153 standalone program or a library component integrated in other software.
154 It generates constrained Delaunay tetrahedralizations, boundary conforming
155 Delaunay meshes and Voronoi partitions. Given a surface mesh, TetGen
156 can tessellate the interior of the domain and preserve the boundary. Steiner
157 points may be added on the boundary for mesh quality. However, a function
158 in TetGen for enforcing that no Steiner points can be added is also available.

159 This is useful for joining two volume meshes sharing the same boundary sur-
160 face. Various constraints and refinement options are provided in TetGen.
161 Briefly, they include:

- 162 1. Minimum dihedral angle and maximum radius-edge ratio for the quality
163 of tetrahedral elements.
- 164 2. Maximum volume and facet area for tetrahedral elements.
- 165 3. User-defined isotropic mesh sizing functions specifying desired edge
166 lengths at any nodal locations. Histograms of mesh quality can be
167 printed.
- 168 4. Adaptive remeshing with respect to newly added nodes. If no nodes are
169 added, the mesh can be reconstructed for mesh refining or coarsening.

170 The input boundary representation (B-Rep) for TetGen is a Piecewise Linear
171 Complex (PLC) (Miller et al., 1996), which is a format of storing the surface
172 mesh for the entire 3D model. A PLC does not have to be a manifold (Lee,
173 2010), indicating that it can contain internal boundaries, which is needed
174 for generating surface-based reservoir models. Each surface in a PLC is
175 associated a unique marker, which can be passed to corresponding triangular
176 facets in the volume mesh for assigning boundary conditions. Regions in a
177 PLC are defined by the coordinates of a node in each region. Each region
178 has a unique integer marker, which can be passed to corresponding elements
179 in the volume mesh for assigning petrophysical properties.

180 3. Adaptive Surface and Volume Mesh Generation

181 The general workflow of building surface-based reservoir models with
182 NURBS curves, Coons patches, adaptive unstructured surface and volume
183 meshes is presented in Fig. 1. Given the open-source libraries, curves and sur-
184 faces can be built using NURBS Toolbox, while tetrahedral volume meshes
185 can be generated by TetGen. The link between 3D surfaces and volume mesh
186 is a surface mesh of the entire model. For each Coons patch, a structured
187 logically Cartesian grid is built. For the four vertical bounding surfaces, un-
188 structured triangular meshes are generated to adapt to internal curves. Then
189 a hybrid surface mesh of the entire model consisting of quadrilaterals and tri-
190 angles as elements is obtained by connecting all structured and unstructured
191 surface meshes.

192 The workflow for building 3D models and mesh generation involves a
193 model builder written in Matlab and a mesh generator in C++. The model

194 builder is based on NURBS Toolbox; the mesh generator is based on Triangle
 195 and TetGen. Structured grids and PLSGs are built in the model builder
 196 and exported to .txt files. The files are subsequently read into the mesh
 197 generator for unstructured surface and volume mesh generation. Triangle
 198 and TetGen are compiled as internal libraries. The inputs for both Triangle
 199 and TetGen are internal. Codes of the model builder and mesh generator
 200 can be downloaded from the linked Mendeley dataset. The detailed steps of
 201 the workflow are as follows.

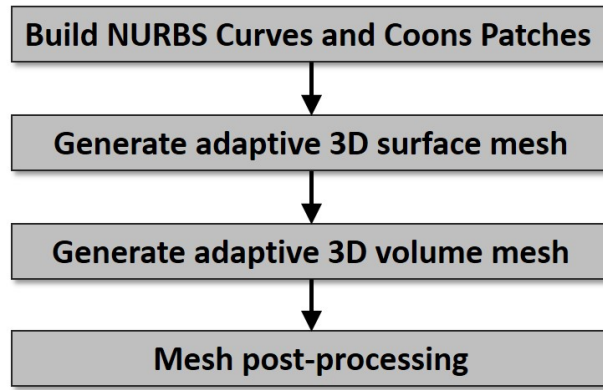


Figure 1: Workflow of building surface-based reservoir models with NURBS curves, Coons-Patches, adaptive unstructured surface and volume meshes.

- 202 1. Design a surface-based reservoir model consisting of NURBS curves
 203 and four vertical bounding surfaces (see Section 5.1).
- 204 2. Build NURBS curves and Coons patches using NURBS Toolbox. Two
 205 Coons patches are not allowed to intersect since it is computationally
 206 expensive to find the exact position of the intersection curve (Abdel-
 207 Malek and Yeh, 1996). Therefore, if two surfaces intersect, we define
 208 a new NURBS curve as the intersection curve, and then build Coons
 209 patches treating the intersection curve as one of the bounding curves.
 210 This is illustrated in Fig. 2.
- 211 3. Build a structured logically Cartesian grid on each Coons patch. Ex-
 212 port all these structured surface grids from the model builder into a
 213 .txt file.
- 214 4. Record NURBS curves on four vertical bounding surfaces. Each verti-
 215 cal surface has two NURBS curves as top and bottom bounding curves,

two vertical left and right bounding lines, and other NURBS curves as internal curves. Then for each vertical surface, discretize all curves into segments. The segment length should be consistent with the structured grid on Coons patches. A PSLG of a vertical surface is obtained by assembling all segments of NURBS curves and segments on the two vertical bounding lines. Export four PSLGs of the four vertical bounding surfaces from the model builder into a .txt file.

5. Read structured surface grids and PSLGs from the .txt files into the mesh generator.
6. Triangle is called by the mesh generator to build a 2D unstructured triangular mesh for each PSLG (the data structure *triangulateio* is used to pass the PSLG into Triangle. Detailed guidance on user controls about mesh quality can be found in Shewchuk (2005)). The 2D coordinates of a PSLG are obtained straightforwardly by ignoring the constant x or y coordinate since each PSLG is on a flat vertical surface (see Appendix A for unstructured mesh generation using Triangle on a 3D curved surface). Then the constant coordinate is added accordingly after mesh generation to obtain a 3D surface mesh. If Steiner points are created on the boundary curves of one surface, they must be treated as additional constraints during mesh generation on neighbouring surfaces such that the discretisation of a curve stays the same on neighbouring surfaces. Therefore, Steiner points are prohibited in the current study to avoid extra computational complexity.
7. Connect structured grids on Coons patches and unstructured meshes on vertical bounding surfaces to obtain a surface mesh in PLC format of the entire model. Then TetGen is called by the mesh generator to build a tetrahedral mesh based on the PLC (the data structure *tetgenio* is used to pass the PLC into TetGen. Detailed guidance on user controls about mesh quality can be found in Si (2013)). Repeating nodes on curves shared between neighbouring surfaces could be retained as TetGen would neglect them automatically. However, self-intersections (e.g. an endnode of an edge lies in the interior of a triangle) are not allowed and can be detected by TetGen.

4. Mesh Post-Processing

The basic data structures of an unstructured tetrahedral mesh include a list of nodes' coordinates and a list of elements' connectivity. The purpose

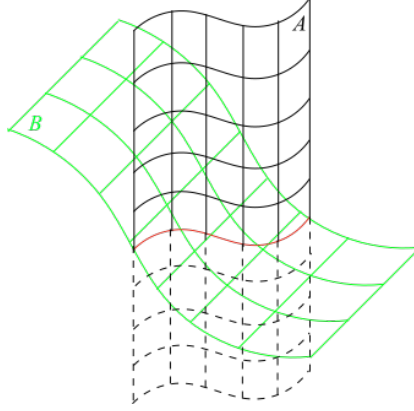


Figure 2: If two surfaces A and B intersect, a new red NURBS curve is defined. Then both A and B are generated as two Coons patches each.

of mesh post-processing is to generate further information in preparation for flow computations. Node-centred finite or control volume methods are widely used for discretisation. An edge-based data structure can be employed to facilitate the implementation of control volume discretisation for partial differential equations (PDE) and reduce computational and storage costs (Lyra et al., 2004; Zhao and Zhang, 2000; Sun et al., 2010; Al Qubeissi, 2013; Akkurt and Sahin, 2017). Mesh post-processing for the edge-based data structure has been implemented in Zhang (2015) and is reviewed here. The post-processing steps mainly include the construction of the following data structures in sequence (Löhner, 2008):

- Elements surrounding nodes: An element surrounds a node if the node is one of the four vertices of the element. The number of elements surrounding each node varies in an unstructured mesh.
- Nodes surrounding nodes: this data structure is built based on elements surrounding nodes. If node i is one of the vertices of elements surrounding node j , then the two nodes are neighbours. The number of nodes surrounding each node varies in an unstructured mesh.
- Edges surrounding nodes: Edges are only built between neighbouring nodes. For each edge, the shared faces between the corresponding control volumes of the two endnodes are computed. The edges-surrounding-nodes structure can simplify the discretisation scheme for unstructured meshes and reduce computational cost.

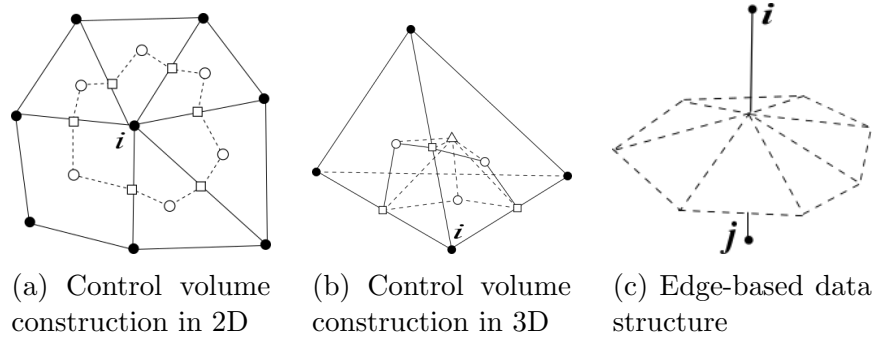


Figure 3: Solid nodes belong to the primary finite element mesh, small circles are centroids of polygons, small squares are edge midpoints, and small triangles represent tetrahedron centroids.

- Build edge-based data structure by grouping shared faces between two control volumes to facilitate discretisation. The implementation is non-trivial and details using pseudocodes can be found in Appendix B.

Figs. 3a and 3b show the control volume construction in 2D and 3D, respectively. In 2D, a segment of the control volume boundary is built between the centroid of a polygon and the midpoint of an edge. Each polygon surrounding node i contains two segments. The control volume of node i is built by connecting all these segments. In a 3D tetrahedral mesh, each tetrahedron surrounding node i contributes six triangular bounding faces for the control volume. The figure shows one of the tetrahedrons. Each triangular bounding face is formed by connecting the centroid of the tetrahedron, a face centroid and an edge midpoint. The control volume of node i in 3D is built by connecting all these bounding faces. The shared faces of control volumes around nodes i and j are grouped to form a small umbrella shown in Fig.3c which is an edge-based data structure.

5. Test Cases

5.1. 3D Model and Volume Mesh Generation

The conceptual geological model is bounded by Coons patches and the four vertical bounding surfaces. Fig. 4 shows the NURBS curves for building Coons patches. These curves represent the intersection between surfaces and cross-sections. Fig. 5 shows the Coons patches built from the curves, where colours correspond to heights (z-coordinates). The Coons patches

296 are represented as structured logically Cartesian grids. Fig. 6 shows the
 297 surface mesh of the entire model. The surface mesh is hybrid and consists of
 298 quadrilaterals and triangles as elements. Unstructured triangular meshes are
 299 built on the four vertical bounding surfaces to adapt to internal intersection
 300 curves.

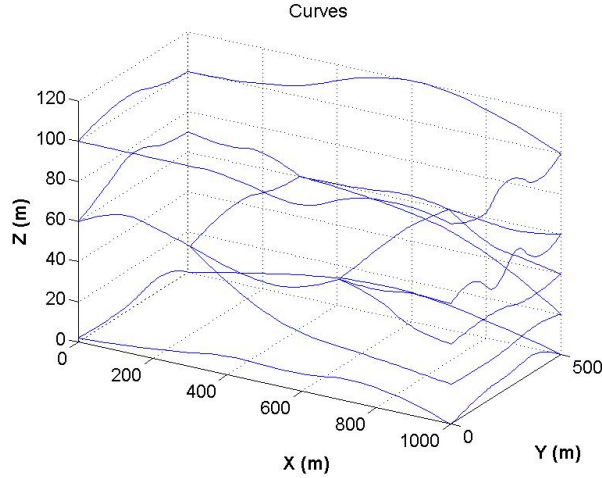


Figure 4: NURBS curves for a geological model. The curves represent the intersection between surfaces and cross-sections.

301 TetGen allows remeshing with respect to newly added nodes. Therefore,
 302 we represent wells as discrete nodes with marks corresponding to wells. The
 303 marks are used to identify nodes at wells after remeshing. In addition, tri-
 304 angular facets on boundaries are identified by associated boundary marks.

305 Fig. 7 shows an unstructured tetrahedral volume mesh built from the
 306 parametric surfaces in Fig. 5. It contains 12610 nodes and 70879 tetrahedral
 307 elements. All bounding surfaces and curves are precisely respected. Geo-
 308 logical regions bounded by surfaces are identified using TetGen by the coor-
 309 dinates of a node inside each region. Taking the four vertical surfaces into
 310 account, the 3D model consists of eleven surfaces bounding four regions. As a
 311 validation example of mesh generation, homogeneous properties are assigned
 312 to each region. The colours in Fig. 7 correspond to horizontal permeability
 313 values. It is usually more robust to generate low-quality volume mesh (e.g.
 314 without constraints on minimum dihedral angle or maximum edge-radius ra-
 315 tio) first, and then refine the mesh to increase quality and adapt to nodes at
 316 wells.

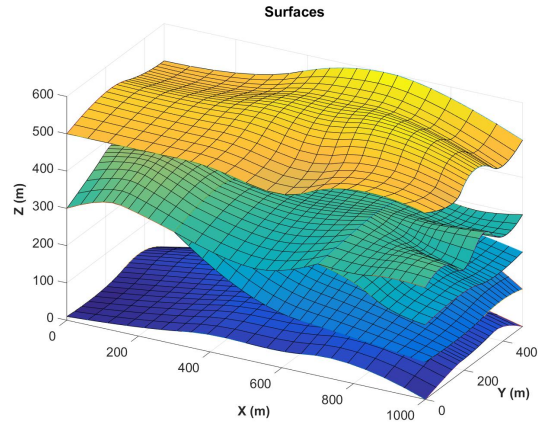


Figure 5: Bilinearly blended Coons patches built from NURBS curves. Colours correspond to heights (z-coordinates). This shows the structured logically Cartesian grids on Coons patches.

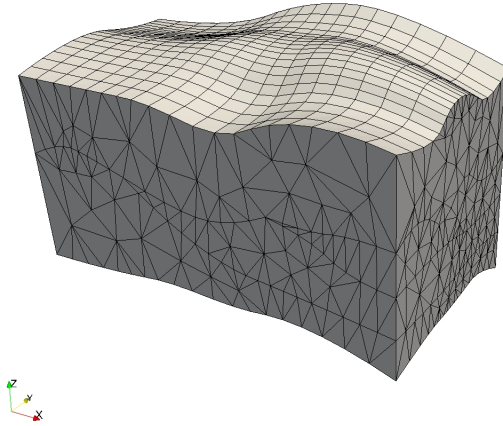


Figure 6: Unstructured hybrid surface mesh in PLC format. Structured grids are built on Coons patches while unstructured grids are on vertical bounding surfaces.

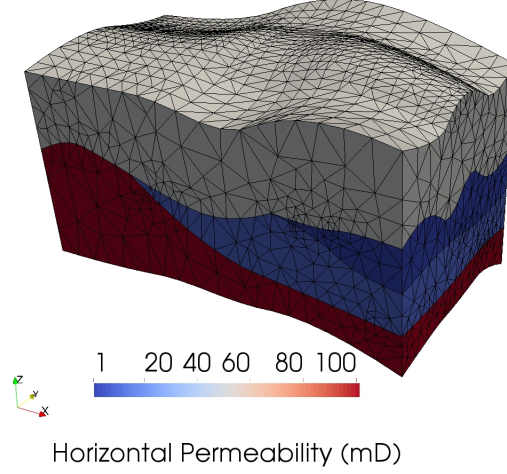


Figure 7: An unstructured tetrahedral volume mesh built from Coons patches. All bounding surfaces and intersection curves are precisely respected. Colours correspond to horizontal permeability values in regions or volumes bounded by surfaces. Steiner points are added on the boundary by TetGen to improve mesh quality.

317 5.2. Two-Phase Flow Immiscible Displacement

318 For validating the volume mesh, two-phase incompressible immiscible dis-
 319 placement is simulated. The governing equations are

$$\frac{\partial \phi S_\alpha}{\partial t} = -\nabla \cdot \vec{u}_\alpha + q_\alpha, \quad \alpha = o, w \quad (4)$$

$$\vec{u}_\alpha = -\frac{k_{r\alpha}}{\mu_\alpha} K(\nabla p_\alpha - \rho_\alpha g \nabla z), \quad \alpha = o, w \quad (5)$$

$$S_w + S_o = 1, \quad (6)$$

$$p_c = p_o - p_w \quad (7)$$

320 where o and w denote oil and water which are the non-wetting and wet-
 321 ting phases, respectively. ϕ is porosity, g is gravitational acceleration, p_c is
 322 capillary pressure and K is the absolute permeability tensor. For phase α ,
 323 p_α is pressure, ρ_α is density, S_α is saturation, \vec{u}_α is velocity, μ_α is dynamic
 324 viscosity, $k_{r\alpha}$ is relative permeability and q_α is volumetric source term. Oil
 325 pressure p_o and water saturation S_w are primary variables. Brooks-Corey
 326 model (Brooks and Corey, 1964) is employed for computing relative perme-
 327 abilities. The residual oil saturation is set to be zero.

$$k_{rw} = \left(\frac{S_w - S_{iw}}{1 - S_{iw}} \right)^4, \quad (8)$$

$$k_{ro} = \left(\frac{1 - S_w}{1 - S_{iw}} \right)^2 \left(1 - \left(\frac{S_w - S_{iw}}{1 - S_{iw}} \right)^2 \right), \quad (9)$$

where S_{iw} is the irreducible water saturation. The control volume finite element method (CVFEM) (Forsyth et al., 1990) is implemented. In CVFEM, pressure is defined on nodes and is piecewise linear in elements, while saturation is piecewise constant in control volumes. CVFEM benefits from both the accuracy of finite element method and the inherent mass conservation of finite volume method. Here, we assemble fluxes between neighbouring control volumes and store them using an edge-based data structure (see Appendix B).

The mesh and the horizontal permeability field are visualised in Fig. 7. The vertical/horizontal permeability ratio is 0.1. Homogeneous porosity is 0.2. Fluid viscosity is 1 cP. Four injectors are placed on the corners that penetrate the entire formation. A curved producer is inside the model. Injectors have Dirichlet condition of pressure equal to 100 bar. The producer has Dirichlet condition of pressure equal to 1 bar. All other boundaries have no-flow conditions.

The histograms of aspect ratio and dihedral angle of all tetrahedrons are shown in Fig. 8. The aspect ratio of a tetrahedron is its longest edge length divided by its smallest side height (Si, 2013). It is reported that mesh quality mainly affects convergence rather than accuracy for the finite element method (Pointwise, 2012). Accuracy is mainly affected by the numerical scheme for simulation given a fixed mesh resolution (Knupp, 2007).

First, we demonstrate CVFEM is convergent and stable on the mesh in Fig. 7. The steady-state pressure solution is shown in Fig. 9. The boundary between high and low permeability has a high gradient of pressure. A cross-section is presented in Fig. 10 to visualise the curved producer with pressure 1 bar. It is obvious that the tetrahedral elements adapt to the nodes of the producer. The initial saturation of the model is at $S_{iw} = 0.2$. Water is injected and oil is produced. The pressure field is kept constant during water flooding. Fig. 11 presents the water saturation field after 347 days. Both pressure and saturation solutions are physically meaningful indicating that the mesh quality is sufficient for stable pressure and saturation solutions in

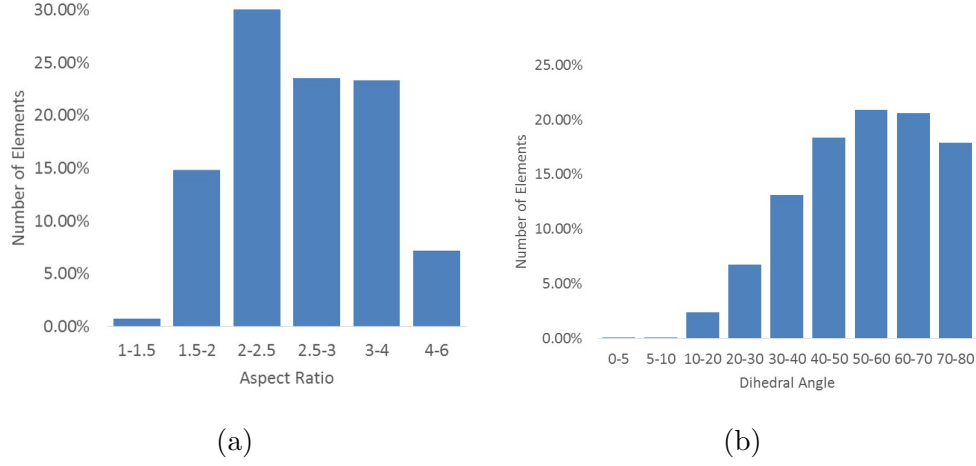


Figure 8: (a) Histogram of aspect ratio for the surface-based reservoir model. (b) Histogram of dihedral angle of the same mesh.

359 this example.

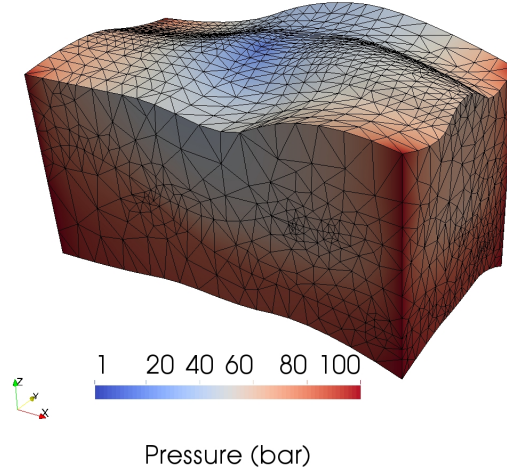


Figure 9: Steady-State pressure field for two-phase immiscible displacement in the surface-based reservoir model. The permeability field is heterogeneous. The boundary between high and low permeability has a high gradient of pressure.

360 Next, the Buckley-Leverett test case is simulated to validate the accuracy
 361 of CVFEM on tetrahedral meshes generated by TetGen against analytical
 362 solution. The model dimension is $100 \text{ m} \times 10 \text{ m} \times 1 \text{ m}$. The histograms of
 363 aspect ratio and dihedral angle of all tetrahedrons in the mesh are exported

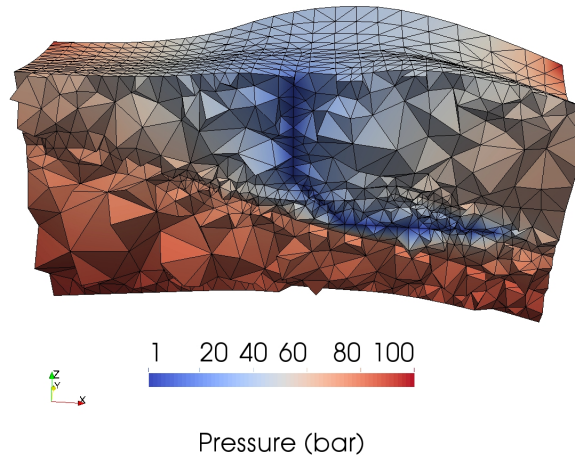


Figure 10: A cross-section to visualise the curved producer with pressure 1 bar. Tetrahedral elements adapt to the well-nodes of the producer.

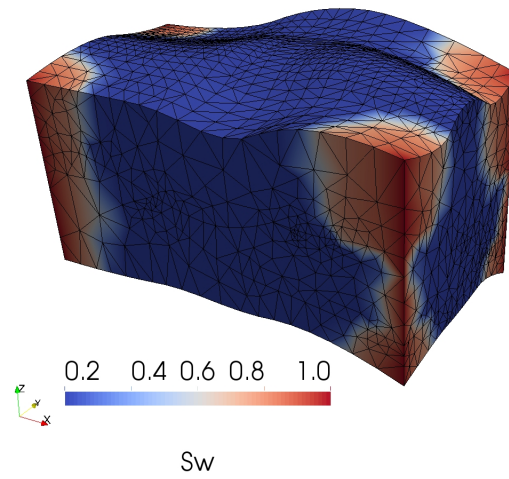


Figure 11: Water saturation field after 347 days for the surface-based reservoir model.

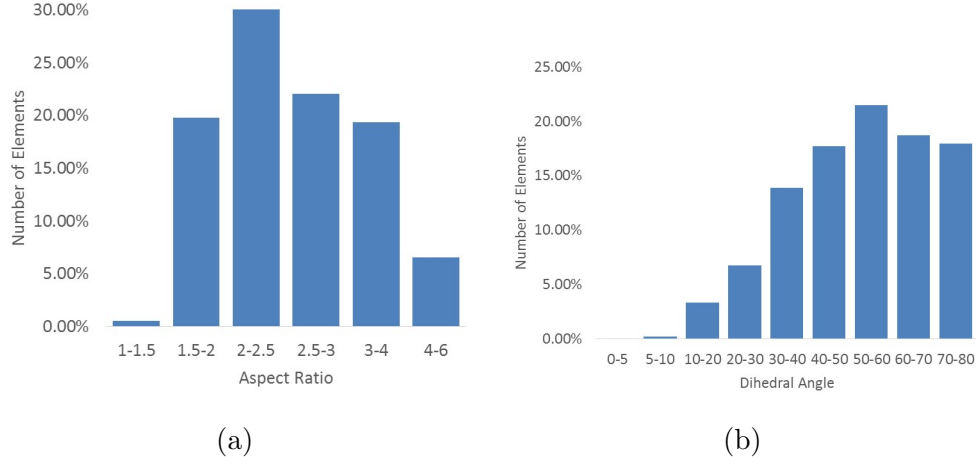


Figure 12: (a) Histogram of aspect ratio for the Buckley-Leverett example. (b) Histogram of dihedral angle of the same mesh.

364 from TetGen and shown in Fig. 12. The mesh quality of both examples is
 365 similar in that the maximum aspect ratio in both cases is below 6 and the
 366 maximum dihedral angle in both cases is below 80 degrees, indicating that
 367 there is no very bad elements (e.g. an element with > 50 aspect ratio).

368 The model is fully saturated with oil initially and water is injected from
 369 the boundary at $x = 0$ with a constant velocity 9.87×10^{-6} m/s to displace
 370 oil. The irreducible water saturation is zero. Homogeneous porosity is 0.2.
 371 Both oil and water viscosities are 1 cP. Gravity and capillary pressure are
 372 neglected. The 3D water saturation field at $t = 5 \times 10^5$ s and the comparison
 373 with analytical result is presented in Fig. 13. The numerical diffusion is
 374 due to the single-point upstream weighting for water fractional flow which
 375 is first-order accurate. The contact front for the finer mesh is sharper as a
 376 consequence of less numerical diffusion.

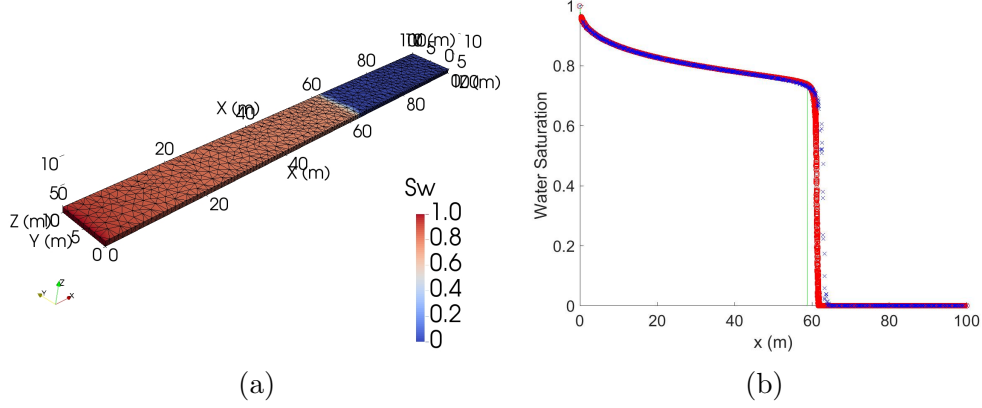


Figure 13: (a) Buckley-Leverett immiscible displacement on an unstructured tetrahedral mesh. (b) Comparison of numerical and analytical saturation profiles. Blue crosses and red circles are numerical results on coarse and fine grids, respectively. The green curve is of analytical result.

6. Conclusions

We have presented a workflow for building surface-based reservoir models using NURBS curves, Coons patches and unstructured tetrahedral meshes. NURBS curves are generated to represent the contacts between surfaces and cross-sections. Parametric surfaces are built based on the curves as Coons patches.

Logically Cartesian structured grids are generated on Coons-patches while unstructured triangular meshes are built on vertical bounding surfaces to adapt to internal intersection curves. The hybrid surface mesh of the entire 3D model consisting of triangles and quadrilaterals is constructed by connecting all meshes on individual surfaces.

The surface mesh is stored in PLC format and TetGen is called to build unstructured tetrahedral meshes. The surfaces bounding geological regions are accurately respected by tetrahedral elements. Further constraints can be applied for high-quality mesh generation. Well configurations in terms of location and geometry are particularly flexible, facilitated by local mesh adaptation.

Control volumes are built based on tetrahedral elements to facilitate CVMFEM discretisation. An edge-based data structure is used to store the vector area and flux between two neighbouring control volumes. The mesh

397 quality and the accuracy of applying CVFEM on unstructured meshes gen-
398 erated by our workflow have been validated by comparing to the analytical
399 solution of the Buckley-Leverett example.

400 In our workflow, all libraries for curve, surface and mesh generation are
401 open-source. The benefits are that first, they are free-of-charge for non-
402 commercial uses; second, we have access to the source codes for deeper un-
403 derstanding and better control while commercial software packages may not
404 be fit-for-purpose; third, Triangle and TetGen can be compiled and linked in-
405 ternally without input or output files such that the workflow becomes more
406 efficient; finally, the libraries can be incorporated in software development
407 under proper licenses.

408 If the bottom-hole pressure (BHP) is of interest, a well model could be ap-
409 plied to couple BHP and well-nodes' pressure (Peaceman, 1978). Fractures
410 will be considered in our future study. The embedded fracture modelling
411 (EDFM) method (Moinfar et al., 2014) will be adopted to avoid the com-
412 putational complexity involved in adapting the volume mesh to fractures of
413 complex geometry.

414 Acknowledgements

415 This paper is from an interesting discussion between authors about using
416 open-source libraries for modelling, meshing and simulation. Our codes are
417 open-source and can be downloaded from the linked Mendeley dataset or
418 by emailing zzhang6666@gmail.com. NURBS Toolbox can be downloaded
419 from [https://uk.mathworks.com/matlabcentral/fileexchange/26390-nurbs-](https://uk.mathworks.com/matlabcentral/fileexchange/26390-nurbs-toolbox-by-d-m-spink)
420 [toolbox-by-d-m-spink](https://uk.mathworks.com/matlabcentral/fileexchange/26390-nurbs-toolbox-by-d-m-spink). Triangle can be downloaded from [https://www.cs.cmu.edu/](https://www.cs.cmu.edu/quake/triangle.html)
421 [quake/triangle.html](https://www.cs.cmu.edu/quake/triangle.html). TetGen can be downloaded from [http://wias-berlin.de/](http://wias-berlin.de/software/tetgen/)
422 [software/tetgen/](http://wias-berlin.de/software/tetgen/). All these codes should be used together to implement the
423 workflow presented in this paper.

424 Bibliography

- 425 Abdel-Malek, K., Yeh, H.-J., 1996. Determining intersection curves between
426 surfaces of two solids. *Computer-Aided Design* 28 (6-7), 539–549.
- 427 Akkurt, S., Sahin, M., 2017. An efficient edge based data structure imple-
428 mentation for a vertex based finite volume method. In: 23rd AIAA Com-
429 putational Fluid Dynamics Conference. p. 3292.
- 430 Al Qubeissi, M., 2013. Development of a conjugate heat transfer solver. LAP
431 LAMBERT Academic Publishing.
- 432 Bentley, M., 2016. Modelling for comfort? *Petroleum Geoscience* 22 (1),
433 3–10.
- 434 Bentley, M., Ringrose, P., 2017. Future directions in reservoir modelling:
435 new tools and fit-for-purpose workflows. In: Geological Society, London,
436 Petroleum Geology Conference series. Vol. 8. Geological Society of London,
437 pp. PGC8–40.
- 438 Borouchaki, H., Laug, P., George, P.-L., 2000. Parametric surface meshing
439 using a combined advancing-front generalized delaunay approach. *Interna-*
440 *tional Journal for Numerical Methods in Engineering* 49 (1-2), 233–259.
- 441 Brooks, R., Corey, T., 1964. Hydraulic properties of porous media. *Hydrology*
442 *Papers*, Colorado State University.

443 Cavero, J., Orellana, N. H., Yemez, I., Singh, V., Izaguirre, E., 2016. Importance of conceptual geological models in 3d reservoir modelling. *first break*
444 34 (7), 39–49.
445

446 Chew, L. P., 1993. Guaranteed-quality mesh generation for curved surfaces.
447 In: *Proceedings of the ninth annual symposium on Computational geometry*. ACM, pp. 274–280.
448

449 Colombero, L., Yan, N., McCormick-Cox, T., Mountney, N. P., 2018. Seismic-driven geocellular modeling of fluvial meander-belt reservoirs using a rule-based method. *Marine and Petroleum Geology* 93, 553–569.
450
451

452 De Berg, M., Van Kreveld, M., Overmars, M., Schwarzkopf, O. C., 2000. Computational geometry. In: *Computational geometry*. Springer, pp. 1–
453 17.
454

455 Farin, G., Hansford, D., 1999. Discrete coons patches. *Computer Aided Geometric Design* 16 (7), 691–700.
456

457 Farrashkhalvat, M., Miles, J., 2003. *Basic Structured Grid Generation: With an introduction to unstructured grid generation*. Elsevier.
458

459 Forsyth, P. A., et al., 1990. A control-volume, finite-element method for local mesh refinement in thermal reservoir simulation. *SPE Reservoir Engineering* 5 (04), 561–566.
460
461

462 Guibas, L., Stolfi, J., 1985. Primitives for the manipulation of general subdivisions and the computation of voronoi. *ACM transactions on graphics (TOG)* 4 (2), 74–123.
463
464

465 Knupp, P., 2007. Remarks on mesh quality. Tech. rep., Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States).
466

467 Kumar, A., Camilleri, D., Brewer, M., et al., 2016. Comparative analysis of dual continuum and discrete fracture simulation approaches to model fluid flow in naturally fractured, low-permeability reservoirs. In: *SPE Low Perm Symposium*. Society of Petroleum Engineers.
468
469
470

471 Lee, J., 2010. *Introduction to topological manifolds*. Vol. 940. Springer Science & Business Media.
472

- 473 Lewis, R. W., Nithiarasu, P., Seetharamu, K. N., 2004. Fundamentals of the
474 finite element method for heat and fluid flow. John Wiley & Sons.
- 475 Löhner, R., 2008. Applied computational fluid dynamics techniques: an in-
476 troduction based on finite element methods. John Wiley & Sons.
- 477 Lyra, P., Lima, R., Guimarães, C., de Carvalho, D., 2004. An edge-based
478 unstructured finite volume procedure for the numerical analysis of heat
479 conduction applications. *Journal of the Brazilian Society of Mechanical*
480 *Sciences and Engineering* 26 (2), 160–169.
- 481 Miller, G. L., Talmor, D., Teng, S.-H., Walkington, N., Wang, H., 1996.
482 Control volume meshes using sphere packing: Generation, refinement and
483 coarsening. *Fifth International Meshing Roundtable*, 47–61.
- 484 Milliotte, C., Matthäi, S., 2014. From seismic interpretation to reservoir
485 model: an integrated study accounting for the structural complexity of
486 the vienna basin using an unstructured reservoir grid. *First Break* 32 (5),
487 95–101.
- 488 Moinfar, A., Varavei, A., Sepehrnoori, K., Johns, R. T., et al., 2014. Develop-
489 ment of an efficient embedded discrete fracture model for 3d compositional
490 reservoir simulation in fractured reservoirs. *SPE Journal* 19 (02), 289–303.
- 491 Novakovic, D., White, C. D., Corbeanu, R. M., Hammon Iii, W. S., Bhat-
492 tacharya, J. P., McMechan, G. A., 2002. Hydraulic effects of shales in
493 fluvial-deltaic deposits: Ground-penetrating radar, outcrop observations,
494 geostatistics, and three-dimensional flow modeling for the ferron sandstone,
495 utah. *Mathematical Geology* 34 (7), 857–893.
- 496 Peaceman, D. W., 1978. Interpretation of well-block pressures in numerical
497 reservoir simulation. *SPE Journal* 18 (03), 183–194.
- 498 Pellerin, J., Botella, A., Bonneau, F., Mazuyer, A., Chauvin, B., Lévy,
499 B., Caumon, G., 2017. Ringmesh: A programming library for develop-
500 ing mesh-based geomodeling applications. *Computers & Geosciences* 104,
501 93–100.
- 502 Piegsl, L., Tiller, W., 2012. The NURBS book. Springer Science & Business
503 Media.

Pointwise, 2012. Accuracy, convergence and mesh quality. Tech. rep., The Connector, Pointwise.

Rogers, D. F., 2000. An introduction to NURBS: with historical perspective. Elsevier.

Ruij, J., Caumon, G., Viseur, S., 2016. Modeling channel forms and related sedimentary objects using a boundary representation based on non-uniform rational b-splines. *Mathematical Geosciences* 48 (3), 259–284.

Schreiner, J., Scheidegger, C. E., Fleishman, S., Silva, C. T., 2006. Direct (re) meshing for efficient surface processing. *Computer graphics forum* 25 (3), 527–536.

Shewchuk, J. R., 1996. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In: *Applied computational geometry towards geometric engineering*. Springer, pp. 203–222.

Shewchuk, J. R., 2002. Delaunay refinement algorithms for triangular mesh generation. *Computational geometry* 22 (1-3), 21–74.

Shewchuk, J. R., 2005. Tetgen user’s manual. University of California at Berkeley.

Si, H., 2013. Tetgen user’s manual. WIAS Technical Report No. 13.

Si, H., 2015. Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software (TOMS)* 41 (2), 11.

Spink, D., 2010. Nurbs toolbox. <https://uk.mathworks.com/matlabcentral/fileexchange/26390-nurbs-toolbox-by-d-m-spink>.

Sun, Z., Chew, J. W., Hills, N. J., Volkov, K. N., Barnes, C. J., 2010. Efficient finite element analysis/computational fluid dynamics thermal coupling for engineering applications. *Journal of turbomachinery* 132 (3), 031016.

Tu, J., Yeoh, G.-H., Liu, C., 2018. *Computational fluid dynamics: a practical approach*. Butterworth-Heinemann.

Wang, Y., Ma, G., Ren, F., Li, T., 2017. A constrained delaunay discretization method for adaptively meshing highly discontinuous geological media. *Computers & Geosciences* 109, 134–148.

- 534 Welsh, M., Mainland, G., 2004. Programming sensor networks using abstract
535 regions. In: NSDI. Vol. 4. pp. 3–3.
- 536 Yin, Z., Ayzenberg, M., MacBeth, C., Feng, T., Chassagne, R., 2015. En-
537 hancement of dynamic reservoir interpretation by correlating multiple 4d
538 seismic monitors to well behavior. Interpretation 3 (2), SP35–SP52.
- 539 Zehner, B., Börner, J. H., Görz, I., Spitzer, K., 2015. Workflows for generat-
540 ing tetrahedral meshes for finite element simulations on complex geological
541 structures. Computers & Geosciences 79, 105–117.
- 542 Zhang, Z., 2015. Unstructured mesh methods for stratified turbulent flows.
543 Ph.D. thesis, Loughborough University.
- 544 Zhao, Y., Zhang, B., 2000. A high-order characteristics upwind fv method
545 for incompressible flow and heat transfer simulation on unstructured grids.
546 Computer methods in applied mechanics and engineering 190 (5), 733–756.
- 547 Zhong, D.-H., Li, M.-C., Song, L.-G., Wang, G., 2006. Enhanced nurbs mod-
548 eling and visualization for large 3d geoengineering applications: an ex-
549 ample from the jinping first-level hydropower engineering project, china.
550 Computers & Geosciences 32 (9), 1270–1282.
- 551 Zienkiewicz, O., Taylor, R., Zhu, J., 2013. The Finite Element Method: Its
552 Basis and Fundamentals. Elsevier, pp. 151–209.

553 Appendix A. Unstructured Triangular Mesh Generation on A Sin- 554 gle 3D Curved Surface

555 Since Triangle meshes 2D surfaces, we obtain the corresponding 2D co-
556 ordinates of points on 3D surfaces, generate 2D meshes and then map them
557 into 3D. This is regarded as an indirect approach for meshing 3D surfaces
558 (Borouchaki et al., 2000; Schreiner et al., 2006). The (u, v) coordinates in
559 the parametrisation of Coons patches in NURBS toolbox could be used as
560 2D coordinates. However, the range of u and v is $[0, 1]$ which does not con-
561 tain information of the dimension nor aspect ratio (longest side divided by
562 shortest side) of 3D surfaces. In consequence, a high-quality triangulation in
563 2D might become skewed and low-quality after being mapped into 3D. To
564 solve the problem, distance coordinates in 2D are defined. For a 3D Coons
565 patch, a logically Cartesian $M \times N$ grid discretising the surface is built to
566 facilitate the computation of distance coordinates. Let $P_{m,n}$ be a node in the
567 structured grid where m and n are along u and v directions, respectively.
568 $1 \leq m \leq M$ and $1 \leq n \leq N$. The distance coordinates of $P_{m,n}$ are defined
569 and computed as

$$du_{m,n} = \sum_{i=2}^m |\overrightarrow{P_{i-1,n}P_{i,n}}|, \quad dv_{m,n} = \sum_{j=2}^n |\overrightarrow{P_{m,j-1}P_{m,j}}|, \quad (\text{A.1})$$

570 that can help preserve the dimension of 3D surfaces. For an arbitrary node
571 with parametric coordinates (u, v) on a surface, its 3D coordinates can be
572 obtained as $f(u, v)$ in NURBS Toolbox using the *nrbeval* function. However,
573 we cannot obtain directly parametric nor distance coordinates for an arbi-
574 trary node with 3D coordinates; we cannot obtain directly 3D coordinates
575 from distance coordinates. For this, a structured triangular connectivity is
576 established by splitting each quadrilateral cell in the $M \times N$ structured grid
577 into two triangles for mapping between coordinate systems. Vertices of these
578 triangles have known parametric, distance and 3D coordinates. Without loss
579 of generality, mapping from distance to 3D coordinates is illustrated. Let Q
580 be a node with distance coordinates (du_Q, dv_Q) and assume Q lies on a tri-
581 angle with vertices $i = 1, 2$ and 3 in the structured grid, the 3D coordinates
582 of Q are approximated by the vertices of the triangle as

$$x_Q = \sum_{i=1}^3 x_i \phi_i, \quad y_Q = \sum_{i=1}^3 y_i \phi_i, \quad z_Q = \sum_{i=1}^3 z_i \phi_i \quad (\text{A.2})$$

where ϕ_i denotes the linear shape function for vertex i . Coordinates of i is (x_i, y_i, z_i) . The values of ϕ_i are evaluated using distance coordinates of node Q and the three vertices of the triangle as (Lewis et al., 2004; Zienkiewicz et al., 2013)

$$\phi_1 = \frac{\begin{vmatrix} du_Q & dv_Q & 1 \\ du_2 & dv_2 & 1 \\ du_3 & dv_3 & 1 \end{vmatrix}}{2 \begin{vmatrix} du_1 & dv_1 & 1 \\ du_2 & dv_2 & 1 \\ du_3 & dv_3 & 1 \end{vmatrix}}, \quad \phi_2 = \frac{\begin{vmatrix} du_1 & dv_1 & 1 \\ du_Q & dv_Q & 1 \\ du_3 & dv_3 & 1 \end{vmatrix}}{2 \begin{vmatrix} du_1 & dv_1 & 1 \\ du_2 & dv_2 & 1 \\ du_3 & dv_3 & 1 \end{vmatrix}}, \quad \phi_3 = \frac{\begin{vmatrix} du_1 & dv_1 & 1 \\ du_2 & dv_2 & 1 \\ du_Q & dv_Q & 1 \end{vmatrix}}{2 \begin{vmatrix} du_1 & dv_1 & 1 \\ du_2 & dv_2 & 1 \\ du_3 & dv_3 & 1 \end{vmatrix}}. \quad (\text{A.3})$$

Fig. A.14 shows a 3D surface generated as a bilinearly blended Coons patch from four bounding curves. The structured grid is 61×21 for visualisation. Fig. A.15a shows constrained Delaunay mesh generation in the parametric space. The triangulation mapped into 3D is shown in Fig. A.15b where triangles are stretched and of low-quality. Fig. A.16a shows constrained Delaunay mesh generation using distance coordinates instead. Fig. A.16b presents the corresponding mesh in 3D where it can be observed qualitatively that the quality of triangular elements are better than that in Fig. A.15b.

To compare mesh quality quantitatively, the histograms of aspect ratio are presented in Fig. A.17. The aspect ratio of a triangle is defined to be its circumradius over twice its inradius. Assuming the edge lengths are a , b and c , the aspect ratio is equal to $abc/8(s-a)(s-b)(s-c)$ where $s = 0.5(a+b+c)$ (Farrashkhalvat and Miles, 2003). Fig. A.17 shows that the mesh quality in Fig. A.16b using distance coordinates is higher than that in Fig. A.15b using parametric coordinates.

Appendix B. Building Edge-Based Data Structure

A matrix *edgev* is defined to identify local ordering of points for elemental edges (Löhner, 2008)

$$\text{edgev}(1 : 6, 1 : 2) = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 1 \\ 1 & 4 \\ 2 & 4 \\ 3 & 4 \end{bmatrix} \quad (\text{B.1})$$

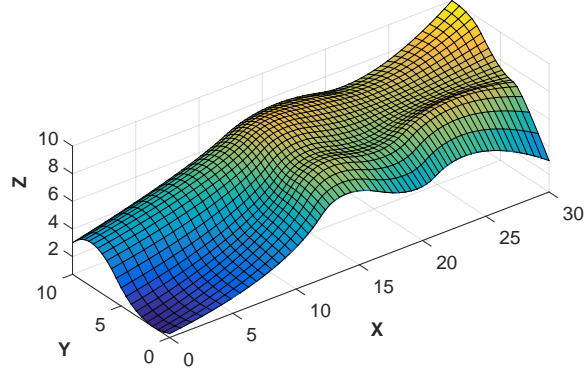


Figure A.14: A 3D surface generated as a bilinearly blended Coons patch from the four bounding curves. Colours reflect heights (z-coordinates).

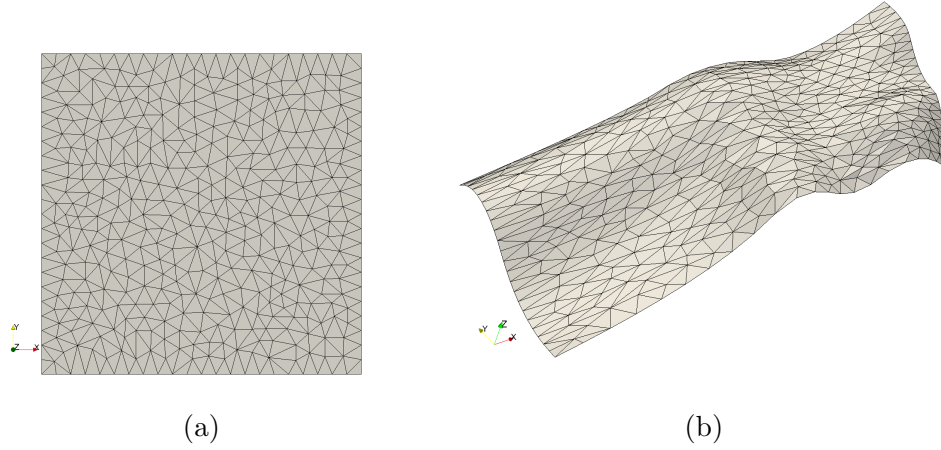


Figure A.15: Constrained Delaunay triangular mesh generation in the parametric space (a) and the corresponding surface mesh in 3D (b).

605 For a tetrahedral element, two triangular faces share an edge. The local
 606 ordering of faces of elemental edges are stored in *edgef*

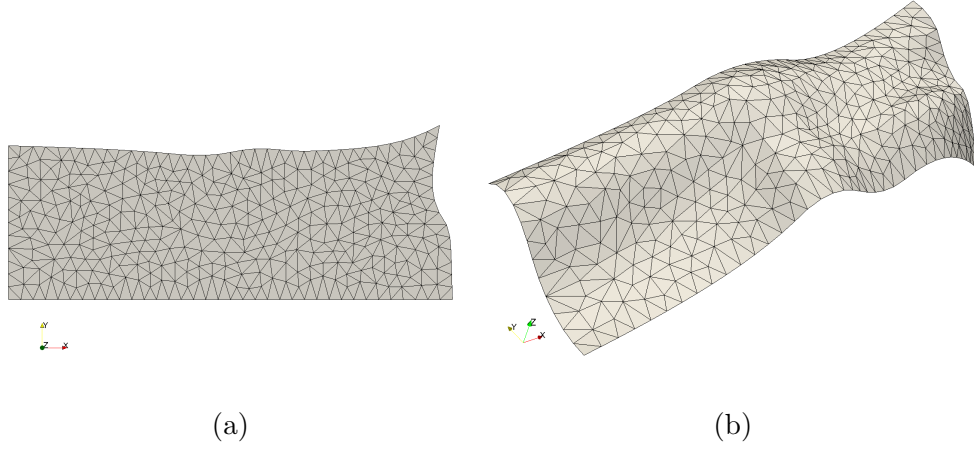


Figure A.16: Triangular mesh generation of the same quality as in Fig. A.15a using distance coordinates in 2D (a) and the corresponding surface mesh in 3D (b). The quality of the surface mesh in 3D is better than in Fig. A.15b

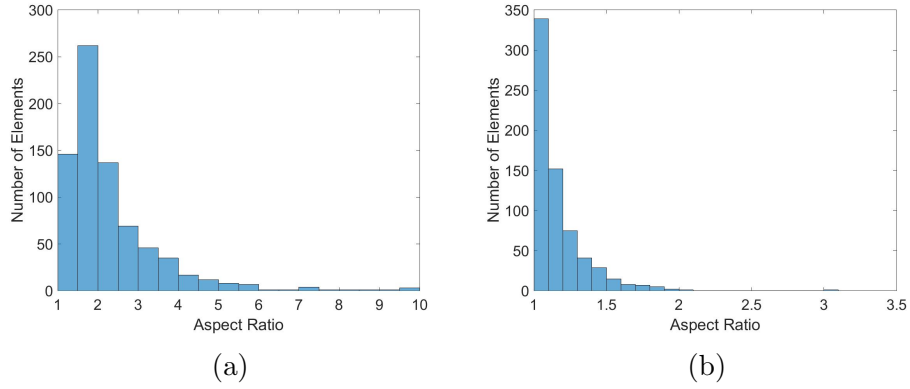


Figure A.17: (a) Histogram of aspect ratio for 3D surface meshes in Fig. A.15b. Around 54% of triangles have aspect ratio less than 2. (b) Histogram of aspect ratio for that in Fig. A.16b. Almost all triangles have aspect ratio less than 2.

$$edgef(1 : 6, 1 : 2) = \begin{bmatrix} 4 & 3 \\ 4 & 1 \\ 2 & 4 \\ 3 & 2 \\ 1 & 3 \\ 2 & 1 \end{bmatrix} \quad (B.2)$$

607 For node-centred finite volume method, vector areas of the shared faces
 608 between control volumes around the two endnodes of each edge are assem-
 609 bled and stored on the edge. For control volume finite element method,
 610 the fluxes through the shared faces are assembled and stored on the edge.
 611 The algorithm based on edges-from-points for computational efficiency. The
 612 pseudocodes are presented here.

```

613 1. //loop over all elements
614 2. FOR iele = 1 ~ nelem
615 3.    //obtain the centroid of element iele
616 4.    POINT point2 = elelist(iele).center
617 5.    //loop over all local edges of element iele
618 6.    FOR i = 1 ~ 6
619 7.        //obtain the IDs of two endpoints of a local edge
620 8.        n1 = elelist(iele).vertex(edgev(i, 1))
621 9.        n2 = elelist(iele).vertex(edgev(i, 2))
622 10.       i1 = min(n1, n2)
623 11.       i2 = max(n1, n2)
624 12.       //loop over all edges from the first endpoint
625 13.       FOR j = 1 ~ pointlist(i1).edgeout.size
626 14.           //obtain the ID of an edge
627 15.           ied = pointlist(i1).edgeout(j)
628 16.           //match the second endpoint
629 17.           IF edgelist(ied).vertex(2) == i2
630 18.               //the ID of the local edge is found
631 19.               iedge = ied
632 20.               BREAK
633 21.           ENDIF
634 22.       ENDFOR
635 23.       //obtain the midpoint of edge iedge
636 24.       POINT point1 = edgelist(iedge).center
637 25.       //vector from the first to second endpoint
638 26.       Define vector  $\vec{a} = \text{pointlist}(i1) \rightarrow \text{pointlist}(i2)$ 
639 27.       //loop over elemental faces connected to the i'th local edge
640 28.       FOR j = 1 ~ 2
641 29.           //obtain the centroid of a elemental face

```

```

642 30.      POINT  $point3 = elelist(iele).face(edgef(i, j)).center$ 
643 31.      //vectors on a shared boundary face for control volumes
644 32.      Define vector  $\vec{b} = point2 \rightarrow point1$ 
645 33.      Define vector  $\vec{c} = point2 \rightarrow point3$ 
646 34.      //normal vector of a shared face
647 35.      Define vector  $\vec{n} = \vec{b} \times \vec{c} / |\vec{b} \times \vec{c}|$ 
648 36.       $sign = 1$ 
649 37.      IF  $\vec{a} \cdot \vec{n} < 0$ 
650 38.           $sign = -1$ 
651 39.      ENDIF
652 40.      //area of shared face
653 41.       $area = 0.5 |\vec{b} \times \vec{c}|$ 
654 42.      //vector area of the face
655 43.       $\vec{S} = \vec{n} \cdot sign \cdot area$ 
656 44.      //assemble vector areas for edge  $iedge$ 
657 45.       $edgelist(iedge).\vec{S} = edgelist(iedge).\vec{S} + \vec{S}$ 
658 46.      //flux through the face
659 47.       $flux = elelist(iele).\vec{u} \cdot \vec{n} \cdot sign \cdot area$ 
660 48.      //assemble fluxes for edge  $iedge$ 
661 49.       $edgelist(iedge).flux = edgelist(iedge).flux + flux$ 
662 50.      ENDFOR
663 51.  ENDFOR
664 52. ENDFOR

```